Need for Reservation Servers with Constrained Deadlines

Daniel Casini, Alessandro Biondi, and Giorgio Buttazzo
Scuola Superiore Sant'Anna, Pisa, Italy
Email: {daniel.casini, alessandro.biondi, giorgio.buttazzo}@santannapisa.it

I. Introduction

So far, all reservation servers have been defined and implemented using an implicit deadline, meaning that the server budget can be consumed anywhere within the server period (see [1], Chapters 5 and 6). However, recent results on semi-partitioned scheduling in multicore systems [2], [3] describe situations in which the server budget has to be consumed within a deadline smaller than the reservation period. Using semi-partitioned scheduling some reservations are statically allocated to processors, whereas others are splitted among multiple processors, thus involving a migration of the workload. When a reservation is partitioned between two cores, its budget Q_i is divided in two portions, $Q_i^{'}$ and $Q_i^{''}$, such that $Q_i = Q_i^{'} + Q_i^{''}$. An efficient method for splitting the budget is the C=D splitting scheme [4], according to which the budget of the second reservation is scheduled with relative deadline $D_i^{''} = Q_i^{''}$, whereas the first one is scheduled with deadline $D_i^{'} = T_i - D_i^{''}$. To apply such a splitting scheme when scheduling reservations, the corresponding servers must be designed to work under constrained deadlines.

This paper focuses on resource reservation in deadline-based systems and formulates the open problem of extending the Hard Constant Bandwidth Server [5] (H-CBS) to work with a deadline less than or equal to its period, in such a way to guarantee a desired reservation bandwidth within a bounded service delay. The H-CBS is based on *earliest-deadline first* (EDF) scheduling and is one of the most used algorithm for implementing resource reservation in deadline-based operating systems, also adopted in the Linux kernel to implement the SCHED_DEADLINE scheduling class [6]. It extends the Constant Bandwidth Server (CBS) [7] by introducing the concept of hard-reservation, according to which, once the budget is depleted, the server is suspended until the next replenishment time. In this manner, the *deadline-aging* problem [8] is avoided and the server execution becomes more regular. One of the main advantages of the H-CBS with respect to other reservation algorithms is to provide a **bounded** *maximum service delay* $\Delta_i = 2(T_i - Q_i)$, which is originated when the worst-case scenario illustrated in Figure 1 occurs. The *maximum service delay* Δ_i and the bandwidth α_i constitute an alternative interface for tuning the reservation parameters (Q_i, T_i). However, the algorithm was designed to work with an implicit-deadline D_i equal to its period T_i.

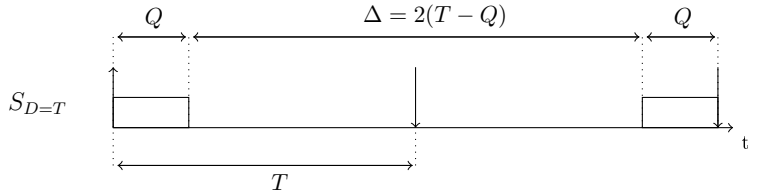

Fig. 1. Maximum service delay considering the H-CBS server, which assumes $D = T$. The worst-case scenario includes two consecutive instances of the server: in the first one, the server provides all its budget at the beginning of the instance, while in the second one the server provides all its budget as later as possible such that the server schedulability is not violated. If workload running upon the server (e.g., a task) arrives after the first instance is completed, then it experiences the maximum service delay of Δ time units.

The H-CBS provides a specific scheduling rule to guarantee the maximum service delay $\Delta_i = 2(T_i - Q_i)$, which deals with the scenario in which the server is idle and new pending workload to be executed arrives. When at time t a job arrives (i.e., the server starts having pending workload) and the H-CBS is idle, a replenishment time is computed as $t_r = d_i - \frac{q_i}{\alpha_i}$, where q_i is the server budget that is available at time t. Then, if $t < t_r$ the server is suspended until time t_r, where the budget is replenished and the absolute deadline is postponed to time $t_r + T_i$; otherwise, the budget is immediately replenished and the absolute deadline is postponed to $t + T_i$. The deadline assignment performed by such a rule exploits the notorious property of EDF scheduling that allows guaranteeing the server schedulability by only reasoning in terms of *bandwidths*.

Assuming constrained deadlines, this H-CBS rule *cannot be used anymore*: indeed, the replenishment time t_r (and the corresponding deadline postponement to $t_r + T_i$) strictly relies on the bandwidth of the server, which—as commonly known—is not enough to ensure the schedulability under EDF in the presence of constrained deadlines. Consequently, the maximum service delay of the worst-case scenario illustrated in Figure 1 (which would be $\Delta_i = T_i + D_i - 2Q_i$) cannot be guaranteed by the current H-CBS scheduling rules.

A naive solution for extending the H-CBS in the presence of constrained deadlines is to discard the budget when the server becomes idle and set the replenishment time as $t_r = kT_i$ with $k \in \mathbb{N}_{\geq 0}$. However, in this way the maximum service delay is

much worse than $T_i + D_i - 2Q_i$. In fact, consider the scenario shown in Figure 2. The server begins its execution in the first period with a pending workload of ϵ units of time, with ϵ arbitrary small. After executing ϵ units of budget, the server becomes idle and discards the budget. Then, immediately after, a new task arrives, but the server has just discarded its budget—so it must wait for the next period. Finally, in the worst-case the server might be scheduled at the end of the following period, thus resulting in a total delay $\Delta_i = T_i + (D_i - Q_i)$. Therefore, the following questions arise:

1) *How we can obtain a new algorithm H-CBS-CD (H-CBS with Constrained Deadline)?*
2) *How to modify the replenishment rules for obtaining a better maximum-service delay?*
3) *Is it possible to achieve a maximum service delay equal to $\Delta_i = D_i + T_i - 2Q_i$?*

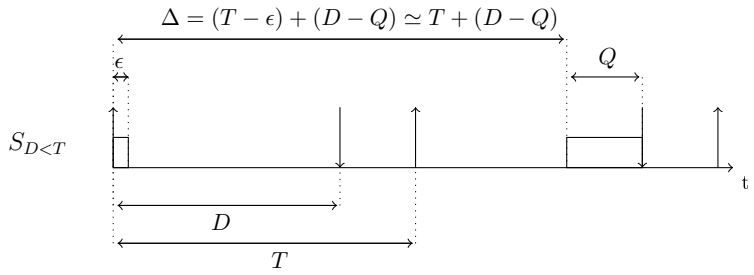

Fig. 2. Maximum service delay considering a naive solution for implementing H-CBS with $D < T$.

Other issues arise when considering resource sharing among tasks scheduled by different reservations. In this case, additional server rules are desirable to avoid budget exhaustions within critical sections, e.g., as done by the BROE [9] server (and in the corresponding version for multiprocessor platforms, M-BROE [10]). BROE extends the H-CBS and employs new scheduling rules that perform deadline postponements based on the server bandwidth: for the same reasons discussed above, such rules cannot be used in the presence of constrained deadlines. Then, another question arises: *4) How to guarantee a constrained-deadline bounded-delay partition in the presence of shared resources?*

Finally, the last issue to consider relies on how to perform admission control for new servers. Indeed, the reservation mechanism is often used in *open environments*, in which software components (implemented by servers) may join the system while the rest of the components continue to operate. Consequently, admission control has to be performed online. Considering implicit-deadline reservations, the admission can easily be done by checking $\sum_{r_i} \frac{Q_i}{T_i} \leq 1$ and then waiting for a safe time at which the new server can be admitted. Conversely, considering constrained deadlines, the schedulability test becomes more complex and will possibly be based on the *processor-demand criterion* (PDC) [11], which has exponential complexity in the worst-case. Then, it results to be unsuitable for being used online. A solution to this problem consists in approximating the PDC, for instance using the *fully polynomial-time approximation scheme* (FPTAS) proposed by Fisher et al. [12]. However, which admission control test is the most suitable for admitting a new reservation has still to be decided; also, it may be the case that a new test must be designed to simplify the on-line scheduling rules related to deadline postponements. Hence, another question arises: *5) Which admission control test should be used for admitting reservations?*

REFERENCES

[1] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, Third Edition*. Springer, 2011.

[2] B. Brandenburg and M. Gul, "Global scheduling not required: Simple, near-optimal multiprocessor real-time scheduling with semi-partitioned reservations," in *Proceedings of the 37th IEEE Real-Time Systems Symposium (RTSS 2016)*, Porto, Portugal, November 29 - December 2 2016.

[3] D. Casini, A. Biondi, and G. Buttazzo, "Semi-partitioned scheduling of dynamic real-time workload: A practical approach based on analysis-driven load balancing," in *Proceedings of the 29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, Dubrovnik, Croatia, 28-30 June 2017.

[4] A. Burns, R. Davis, P. Wang, , and F. Zhang, "Partitioned EDF scheduling for multiprocessors using a C=D task splitting scheme," *Real-Time Systems*, vol. 48, pp. 3–33, 2012.

[5] A. Biondi, A. Melani, and M. Bertogna, "Hard constant bandwidth server: Comprehensive formulation and critical scenarios," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, Pisa, Italy, June 18-20 2014.

[6] J. Lelli, C. Scordino, L. Abeni, and D. Faggioli, "Deadline scheduling in the Linux kernel," *Software: Practice and Experience*, vol. 46, no. 6, pp. 821–839, 2016.

[7] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS 1998)*, Madrid, Spain, December 2-4 1998.

[8] L. Marzario, G. Lipari, P. Balbastre, and A. Crespo, "Iris: A new reclaiming algorithm for server-based real-time systems," in *10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004)*, Toronto, Canada, May, 25-28 2004.

[9] M. Bertogna, N. Fisher, and S. Baruah, "Resource-sharing servers for open environments," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 202–219, July 2009.

[10] A. Biondi, G. C. Buttazzo, and M. Bertogna, "Supporting component-based development in partitioned multiprocessor real-time systems," in *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS 2015)*, Lund, Sweden, July 8-10 2015.

[11] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-time systems*, vol. 2, no. 4, pp. 301–324, 1990.

[12] N. Fisher, T. P. Baker, and S. Baruah, "Algorithms for determining the demand-based load of a sporadic task system," in *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*, Sydney, Australia, Aug, 16-18 2006.